AMENDMENTS TO THE CLAIMS

Please amend the claims as indicated in the following listing of all claims:

1. (Currently Amended) A non-blocking concurrent shared object representation comprising:

a linked-list of nodes encoding of a group of zero or more values; and

linearizable operations defined to implement semantics of at least insert and remove operations on the group, wherein concurrent execution of the linearizable operations is mediated using a first synchronization primitive to encode a marked node indication signifying logical deletion of a corresponding one of the values from the group, wherein the first synchronization primitive atomically examines and updates a single target, the updating being conditional on the examination.

2. (Original) The non-blocking concurrent shared object representation of claim 1, wherein concurrent execution of the linearizable operations is further mediated using a second synchronization primitive to physically excise the node corresponding to the logically deleted value.

3. (Currently Amended) The non-blocking concurrent shared object representation of claim 2,

wherein the first and second synchronization primitives include ~~are~~ compare and swap (CAS) operations and load lock/store-and-compare operations.

4. (Original) The non-blocking concurrent shared object representation of claim 2, wherein reclamation of storage associated with the excised node is independent of the linearizable operations.

5. (Original) The non-blocking concurrent shared object representation of claim 1, wherein the linked-list of nodes is free of reference count storage for coordination of garbage collection.

6. (Original) The non-blocking concurrent shared object representation of claim 1, wherein traversal of the concurrent shared object is without atomic update of a garbage collection coordination store.

7. (Original) The non-blocking concurrent shared object representation of claim 1, wherein successful completion of an insertion into the group requires, at most, one atomic update of the concurrent shared object;
wherein successful completion of a deletion from the group requires, at most, two atomic updates of the concurrent shared object; and
wherein mere traversal of the concurrent shared object is without atomic update of the concurrent shared object.

8. (Original) The non-blocking concurrent shared object representation of claim 1, wherein the node corresponding to the logically deleted value is physically excised from the linked-list by an execution sequence corresponding to one of:
an instance of the remove operation that performed the logical deletion;
an instance of the remove operation that did not perform the logical deletion; and
an instance of the insert operation.

9. (Original) The non-blocking concurrent shared object representation of claim 1, wherein the linearizable operations further implement semantics of a find operation.

10. (Original) The non-blocking concurrent shared object representation of claim 1, wherein the values of the group are stored in respective ones of the nodes.

11. (Original) The non-blocking concurrent shared object representation of claim 1, wherein the values of the group are represented in storage reachable from respective ones of the nodes.

12. (Original) The non-blocking concurrent shared object representation of claim 1, wherein the values of the group are represented in storage identified by respective ones of the nodes.

13. (Original) The non-blocking concurrent shared object representation of claim 1, wherein the marked node indication includes a distinguishing pointer value.

14. (Original) The non-blocking concurrent shared object representation of claim 1, wherein the marked node indication includes a distinguishing bit value in an otherwise unused portion of a next node pointer of the logically deleted node.

15. (Original) The non-blocking concurrent shared object representation of claim 1, wherein respective next node pointers of those of the nodes corresponding to current values of the group directly reference respective other ones of the nodes; and wherein the marked node indication includes a distinguishing additional level of indirection between the next node pointer of the logically deleted node and a respective other one of the nodes.

16. (Currently Amended) A method of managing access to a linked-list of nodes susceptible to concurrent operations on a group encoded therein, the method comprising:

separating deletion of a value from the group into at least two functional sequences;

the first functional sequence performing a logical deletion of the value using a synchronization primitive to mark a corresponding one of the nodes; and

the second functional sequence excising the marked node from the linked-list,

wherein the synchronization primitive atomically examines and updates a single target, the updating being conditional on the examination.

17. (Original) The method of claim 16,

wherein the logical deletion and the marked node excision are performed as part of a single deletion operation operating upon the value.

18. (Original) The method of claim 16,

wherein the logical deletion is performed as part of a deletion operation operating upon the value; and

wherein the marked node excision is performed as part of another operation.

19. (Original) The method of claim 18,
wherein the another operation is an insert operation.

20. (Original) The method of claim 18,
wherein the another operation is a remove operation operating upon another node.

21. (Currently Amended) The method of claim 16,
wherein the synchronization primitive includes a compare and swap (CAS) operation or a load lock/store-and-compare operation.

22. (Original) The method of claim 16, further comprising:
after the logical deletion but before the marked node excision, traversing, as part of an access operation, the linked-list including the marked node.

23. (Original) The method of claim 16,
wherein the group is an ordered set; and
wherein the linearizable operations implement semantics of a remove operation selective for a value, if any, of the group based on comparison with a specified value.

24. (Original) The method of claim 23,
wherein the ordered set is organized in increasing value order; and
wherein the remove operation is selective for a value, if any, of the group greater than or equal to the specified value.

25. (Currently Amended) A computer program product encoded in at least one computer readable medium, the computer program product comprising:
at least two functional sequences providing non-blocking access to a concurrent shared object, the concurrent shared object instantiable as a linked-list of nodes encoding of a group of zero or more values;
a first of the functional sequences defined to implement semantics of an insert operation on the group; and

a second of the functional sequences defined to implement semantics of a remove operation on the group,

wherein instances of the functional sequences are linearizable and concurrent execution thereof by plural processors of a multiprocessor is mediated using a synchronization primitive to encode a marked node indication signifying logical deletion of a corresponding one of the values from the group with separate physical excision of the corresponding node, wherein the synchronization primitive atomically examines and updates a single target, the updating being conditional on the examination.

26. (Original) A computer program product as recited in 25,

embodied as a software component combinable with program code to provide the program code with linearizable, non-blocking access to the concurrent shared object.

27. (Original) A computer program product as recited in 25,

embodied as a program executable to provide linearizable, non-blocking access to the concurrent shared object.

28. (Original) A computer program product as recited in 25,

wherein the at least one computer readable medium is selected from the set of a disk, tape or other magnetic, optical, or electronic storage medium and a network, wireline, wireless or other communications medium.

29. (Currently Amended) An apparatus comprising:

plural processors;

one or more data stores addressable by each of the plural processors;

means for coordinating concurrent execution, by ones of the plural processors, of at least insert and remove operations on a group of zero or more values encoded in the one or more data stores, the coordinating employing a first synchronization primitive to encode an indication signifying logical deletion of a corresponding one of the values from the group and a second synchronization primitive to

physically excise the node corresponding to the logically deleted value, wherein the synchronization primitives atomically examine and update their respective single targets, the updating being conditional on the examination; and

means for traversing the encoded group without use of an atomic operation.

30. (New) The apparatus of claim 29 wherein the synchronization primitives include a compare-and-swap operation and a lock load/store-and-compare operations